TP2 MI11/AI39 Linux embarqué

Semestre printemps 2025

Guillaume Sanahuja – guillaume.sanahuja@hds.utc.fr

Table des matières

Préambule

Pour ce TP, vous devez réutiliser la machine virtuelle de la séance précédente.

Un compte rendu de TP au format PDF est à rendre sur le moodle de l'UV. Il vous est demandé d'y écrire les réponses aux questions figurant dans les différents exercices. Également, vous pouvez compléter le compte rendu avec toute information que vous jugerez utile, par exemple les manipulations que vous avez réalisées pendant la séance, les résultats que vous avez observés, etc.

Travail préalable à la séance de TP

Relire le cours sur Linux embarqué disponible sur le moodle. Prendre connaissance de la *datasheet* du Joy-Pi-Note disponible sur le moodle, et des caractéristiques de la carte RaspberryPi 4. Avoir compris le précédent TP.

Exercice 1 : Hello World

Maintenant que nous avons pu cross-compiler notre système et le démarrer, nous allons programmer un premier exécutable qui affiche le message « Hello World ! ». Etant donné que le code sera inclus uniquement dans un seul fichier source, nous n'avons pas besoin d'utiliser de

Makefile.

Créez un fichier hello.*c* et éditez le. Ajoutez le code C qui permet d'afficher le message voulu puis compilez votre programme avec la commande suivante :

```
gcc hello.c -o hello
```

Utilisez la commande *file* pour obtenir des renseignements sur le fichier *hello.*

Question 1.1 : Que constatez vous ? Pourquoi ce fichier ne s'exécutera pas sur la cible ?

Cross-compilez maintenant votre programme avec la commande suivante :

```
$CC hello.c -o hello
```

Question 1.2 : Que faut-il faire avant de pouvoir lancer cette commande ?

Question 1.3 : Utilisez de nouveau la commande file, que constatez vous ?

Vérifiez ensuite le bon fonctionnement sur la cible.

Question 1.4 : Fournissez le code dans le compte rendu et le résultat obtenu dans le terminal.

Exercice 2 : Clignotement des leds

Nous allons maintenant manipuler les périphériques de la cible. Pour que ce soit visuel, nous allons nous attaquer aux traditionnelles leds. Vous utiliserez les deux leds définies lors de la séance de TP précédente.

Question 2.1 : Rappelez comment accéder aux leds en manipulant des fichiers.

Dans le terminal, affichez les valeurs de ces fichiers et modifiez-les.

Question 2.2 : Fournissez les 4 lignes de commandes permettant d'allumer et d'éteindre les 2 leds.

Créez maintenant un nouveau fichier *led.c* qui devra allumer en alternance les 2 leds chaque seconde. Compilez ce code avec la chaîne de compilation croisée et testez-le sur la cible. Pour rappel, les entrées sorties étant vues comme des fichiers sous Linux, vous pouvez utiliser les fonctions open, read, write et close pour y accéder et contrôler les leds.

Question 2.3 : Fournissez le code source led.c et faites valider le fonctionnement de l'application par le chargé de TP.

Exercice 3 : Boutons poussoirs

Les boutons poussoirs du joypinote sont accessibles via le fichier */dev/input/event0.* Faites un essai de lecture de l'état des boutons en ligne de commande avec le programme *evtest*.

Question 3.1 : Donnez la commande utilisée. Quelles sont les valeurs des différents événements ?

Écrivez maintenant un programme réagissant aux actions sur les boutons, en affichant un message et/ou allumant une led par exemple. Pour cela, vous devez stocker le résultat de la lecture du fichier dans une structure de type *input_event*, qui contiendra donc les informations liées à l'événement d'une touche.

Question 3.2 : Faites une recherche dans les fichiers headers où est installée la cross-toolchain pour trouver où est déclarée la structure input_event.

Cross-compilez votre programme et testez le.

Question 3.3 : Fournissez le code source et faites valider le fonctionnement de l'application par le chargé de TP.

Exercice 4 : Charge CPU

Cet exercice va vous permettre de vérifier l'incidence de la charge CPU sur une tâche périodique. Ecrivez donc un programme réalisant 10 000 fois une attente de 1ms. Utilisez la fonction *clock_gettime* associée à la clock *CLOCK_REALTIME* pour mesurer le temps total. Utilisez maintenant la commande *stress* pour charger le CPU.

Question 4.1 : Quels sont les différents temps relevés ? Faites plusieurs essais avec des valeurs sensiblement différentes du stress.

Améliorez votre programme afin de calculer et d'afficher les latences minimum, maximum et moyenne de chaque réveil suite à la fonction d'attente de 1ms. Certaines de vos variables devront être stockées dans des *uint64_t* afin d'éviter les débordements. L'affichage de ce type de données s'effectue avec *%lld* dans la fonction *printf*.

Question 4.2 : Quels sont les différents temps relevés ? Quelles sont vos conclusions ? Comment pourrait-on améliorer les résultats ?

Question 4.3 : Fournissez le code source de votre programme.

Exercice 5 : Joystick et écran LCD

Le but de cet exercice est d'écrire un programme réagissant au *joystick* afin de changer l'affichage de l'écran LCD.

Question 5.1 : Sachant que le joystick est un périphérique d'entrée tout comme les boutons poussoirs, quel est le fichier à manipuler ?

Faites un essai de lecture de l'état du *joystick* en ligne de commande.

Question 5.2 : Quelles sont les caractéristiques du joystick?

L’écran LCD est accessible via le fichier /*dev/lcd*. Faites un essai de modification de l’affichage du LCD en ligne de commande.

Question 5.3 : Donnez la commande utilisée.

Codez maintenant un programme affichant sur une ligne la valeur x du *joystick*, et sur la seconde ligne la valeur y.

Question 5.4 : Fournissez le code source et faites valider le fonctionnement de l'application par le chargé de TP.